ISSN 0956-9979 DECEMBER 1995

VIRUS BULLETIN

THE INTERNATIONAL PUBLICATION ON COMPUTER VIRUS PREVENTION, RECOGNITION AND REMOVAL

IN THIS ISSUE:

• **Worms and other animals.** Viruses are not the only invasive threat faced by the computing community – worms also play their parts in the overall picture. For a description of the threat, turn to p.15.

• **Black Baron sentenced.** On 17 November 1995, Christopher Pile was sentenced to 18 months in prison. He pleaded guilty to eleven charges under the *Computer Misuse Act*: details on p.3.

• **Editors and interviews.** Until earlier this year, Fridrik Skulason was *VB's* technical editor. Now, find out what he does apart from dissecting viruses – see p.7.

CONTENTS

EDITORIAL

Reach out and touch someone

“ this new generation of Internet tools will bring multiple platform viruses and worms a real possibility ”

The Internet is a security risk. Well, big deal, film at eleven – such a statement is equivalent to saying something like ‘leaving your front door open may mean you run the risk of being burgled’. Is this news? It could be, in a fairly short time. Strange things are afoot on the information bridle-path – sorry, superhighway.

Firstly, and most obviously, we have the growth factor. There is no accurate way to determine how many ‘users’ the Internet has, but there is no arguing with the fact that it is growing at a rate close to exponential. What was once simply a chattering family of academics exchanging everything from tips on cold fusion to recipes for cookies is now a fully-fledged community, with all the extremes that that entails. Any one of them can knock on your metaphorical door, and, using one of a fairly limited number of keys, stands a reasonably high chance of breaking in.

That the Internet is being used more and more encourages the development of simpler ways to use it; simpler as far as the user is concerned, but far more complex under the surface than what has gone before. We are witnessing major steps in the development of the Internet – the WWW has in some ways superseded more primitive tools such as telnet and FTP, and in the future so-called ‘intelligent software agents’ and the like will come to the fore.

Sun Microsystems, a company which has always been at the forefront of information technology, is in the course of rolling out *HotJava* and *Java* – a web browser and programming language respectively. The two are inextricably intertwined (*HotJava* allows the automatic downloading and execution of *Java* ‘applets’, extensions to the browser itself). Want to view a TIFF file? If your *HotJava* browser doesn’t support TIFF, no big deal; a *Java* applet can be downloaded and automatically installed, allowing you to view such images. The system is platform-independent: the applets are not downloaded as human-readable *Java* source-code, but as ‘bytecode’; essentially, code which is run in the *HotJava* environment. The user need never be aware that any of this has happened. What an opportunity.

Sun states that the system is secure, and documents numerous internal systems through which incoming bytecode has to pass to prevent untoward things happening within your browser, including special attention to preventing incoming messages over-running data areas within the browser (‘spamming the buffer’). It was this type of attack which permitted both the Internet Worm of 1988, and the much more recent attack on the *NetScape* browser.

Nevertheless, this type of system must offer viral opportunities. Multi-platform viruses have been brought to our attention recently by macro viruses – it is inevitable that this new generation of Internet tools will bring multiple platform viruses and worms a real possibility.

Turning away from *Java* for a moment, I focus my attention on the *Microsoft Network* (*MSN*), the online service which caused such a hoo-hah recently because the access software is bundled with (indeed, integral to) *Windows 95*. This network provides the ability to encapsulate executables inside email messages in such a way that they execute automatically on the recipient’s computer.

In this way even conventional viruses can be made to hit PCs; no further effort required. Coupled with the growth of unsolicited email (more often than not trying to sell you something) and the posting of messages to large numbers of inappropriate newsgroups and/or mailing lists (curiously, this latter is also known as ‘spamming’), this could leave users of the *MSN* at risk. Not, however, that much more than users of MIME (Multi-purpose Internet Mail Extensions), which allows executables to be translated automatically into a state suitable for transmission over the Internet. Some of the more exotic email systems allow automatic execution of such attachments, placing users of such systems in a similar position to those of the *MSN*.

All of this goes to show what is so often true – ease of use also implies ease of misuse. And the Internet is steadily becoming easier to use…

NEWS

Black Baron Behind Bars

At Exeter Crown Court on 15 November 1995, Christopher Pile (alias Black Baron) appeared for sentencing for eleven offences under Sections 2 and 3 of the *Computer Misuse Act*. Pile was first arrested sixteen months ago, and had already pleaded guilty to all offences at a hearing at Plymouth Crown Court in May.

Testimony from Jim Bates, the *Crown Prosecution Service's* (*CPS*) expert witness, was instrumental to the Prosecution's case. Counsel for the Defence, subsequent to Pile entering guilty pleas at the May hearing, asked for an adjournment in order to call their own expert witness prior to sentencing.

The indictment provided ten specimen counts of Pile's activities relating to unauthorised access to computers and unauthorised modification of data. The eleventh, regarded by the *CPS* as the most serious, was knowingly inciting other people to cause unauthorised modification to computer programs and data; i.e. writing viruses.

The last charge related to an incident on 22 June 1994, when Pile uploaded to the *Abbey BBS* the file SMEG03.ZIP. This included an on-line 'training manual', plans for the encryption engine, and the training viruses Germ and Trivia, with which users could experiment.

Pile wrote various pieces of code for distribution, including SMEG (the Simulated Metamorphic Encryption Generator) and the viruses Pathogen and Queeg. These he uploaded to BBSs in the UK, hiding them in various files and utilities.

Users and companies across the UK downloaded programs and became infected: *Apricot Computers* and *Mapline Engineering* estimated that exposure to Pathogen cost each in the region of £1000, and *Microprose*, which shut down its networks both in the UK and the USA for three weeks after an attack of Pathogen, put its total costs at over £250,000.

Pile's first distributed virus, Pathogen, was released with SMEG v0.1. When the virus reaches its 32nd generation, it triggers on Mondays between 17:00 and 17:59, erasing random sectors on the first physical disk. It displays the message: 'Smoke me a kipper, I'll be back for breakfast… Unfortunately, some of your data won't'.

This message, asserted Prosecution lawyer Brian Lett, indicated clearly that Pile *was* aware that the virus would cause damage. The payload's timing, it was further claimed, was set to cause maximum damage, it being a time when most office workers would be at work with their systems.

Queeg, the second virus Pile released, was encrypted with, SMEG v0.2. Pile had by this time refined the engine, making Queeg more difficult to detect than Pathogen, and had changed the timing of the trigger to Sundays at 13:00.

Prevalence Table – October 1995

Virus	Incidents	(%) Reports
Form.A	36	13.1%
AntiEXE.A	26	9.5%
Empire.Monkey.B	22	8.0%
Parity_Boot.B	20	7.3%
Ripper	20	7.3%
Concept	17	6.2%
AntiCMOS	16	5.8%
BuptBoot	10	3.6%
Junkie	7	2.5%
EXEBug.A	5	1.8%
Jumper.B	5	1.8%
Manzon	5	1.8%
One_Half	5	1.8%
Stoned.Angelina	5	1.8%
Cascade.1701	4	1.5%
J&M	4	1.5%
Kampana	4	1.5%
NYB	4	1.5%
Tequila	4	1.5%
She_Has	3	1.1%
V-Sign	3	1.1%
WBoot.A	3	1.1%
Boot.437	2	0.7%
Die_Hard	2	0.7%
Natas	2	0.7%
Quicky.1376	2	0.7%
Rhubarb	2	0.7%
Sampo	2	0.7%
StealthBoot.C	2	0.7%
Stoned.Dinamo	2	0.7%
Stoned.Manitoba	2	0.7%
Stoned.Kiev	2	0.7%
Tai-pan.438	2	0.7%
Other *	25	9.1%
Total	275	100%

* The Prevalence Table includes one report of each of the following viruses: Astra.927, Barrotes, Boot.B, Byway.A, Chameleon, Crazy_Boot, Crazy_Nine, Empire.Monkey.A, Espejo, FITW.7918, Frank, Green_Caterpillar.1575, HLL.Halloween, Jerusalem.Standard, K-Hate, Kilroy, Phx.965, Stoned.NoInt, Stoned.Sepultura.A, Stoned.Wonka, Swiss_Boot, Tai-pan.666, Tremor, Unashamed, and WelcomB.

At his first interview, Pile not only denied being the author of the viruses, but claimed to have no technological knowledge of computers whatever. After speaking with his solicitor, Pile admitted being the author and distributor of Pathogen, Queeg, and SMEG, but always maintained that the prime object of the exercise was not to distribute viruses, but to distribute SMEG, and claimed that SMEG could be used for applications other than virus creation.

During his final interviews, Pile expressed remorse and regret for his actions, saying that he wished he could 'turn the clock back'. Counsel for the Defence portrayed Pile as a sad figure of a man; a loner with no friends, who never wanted anything to do with computers again. Pile is said to have told his counsel that 'a prison term is a state of mind which I have been serving for 16 months'.

The Outcome

On handing down sentence, Judge Jeremy Griggs said the clock could not be turned back; that it was impossible to determine the extent of the damage which might still happen. The maximum sentence for an offence of this type would be five years, Griggs said, but such a sentence would only be appropriate for a person who carried out such actions with intent of personal gain. Griggs was satisfied that this was not the case here.

Had Pile pleaded not guilty, the sentence would have been in the region of three years; however, in view of his cooperation with the police, amongst other considerations, he was sentenced to eighteen months' detention. Furthermore, all his computer equipment and software was to be confiscated under Section 73 of the 1973 *Powers of Criminal Court Act*.

Detective Constable John Samuels, of the Devon and Cornwall Police, said of the case: 'It is a good result. It is the first prosecution of its kind here, and has resulted in a custodial sentence. This must show other virus writers that, should they indulge in similar activities, they also can expect to go to prison.'

Two significant points of the conviction, said Samuels, related to specific charges. First, an offence under Section 3, in which a virus which had been in the public domain and infected a user's machine. The police could not prove the link between that system and Pile; however, it was considered sufficient to have proven that Pile had written the virus. Second, the final charge; that of inciting others to write viruses – once again, a warning to others that by encouraging others to commit a crime, they may in fact be committing a crime themselves ∎

UTR, the Folding Newsletter

It is with interest that *Virus Bulletin* notes the passing into the annals of history of the *Underground Technology Review* (*UTR*). *UTR* was a monthly newsletter published by Mark Ludwig of *American Eagle Inc* which replaced *Computer Virus Developments Quarterly* at the end of 1994. Although *UTR* did not concentrate solely on computer viruses, a fair proportion of its content did.

Ludwig, author of *The Little Black Book of Computer Viruses*, stated in the final issue of the *UTR* that publication of the newsletter is ceasing because of lack of readership. Other factors include the financial difficulties associated with the drop in cover price and increase in expenditure caused by going monthly – a venture described by Ludwig as 'a big mistake'.

In an industry with so many shades of grey, Ludwig's publications have always been at the darker end of the scale: this is a trend which seems set to continue. His latest announcement concerns his intention to write a new computer virus tome, which will be called *Computer Virus Supertechnology 1996*, which will be offered to current *UTR* subscribers at discounted rates.

Ludwig states in *UTR* that the book 'will contain some cutting edge computer viruses for the 1990s' and 'is not going to be for the lame-of-heart, the hesitant, or the whiner'. Fortunately, the high price of the book (described as 'around $100') may well prevent it from reaching a very wide audience ∎

VB '96 – the Cycle Begins Again

Mere weeks after *VB* returned en masse from the USA after *VB '95*, preparations for next year's conference (imaginatively titled *VB '96*) are well under way. Next year's event will be held at the *Grand Hotel* in Brighton, UK, on 19/20 September 1996.

Brighton, a traditional seaside retreat, is just an hour from Central London and thirty minutes from London's Gatwick Airport. The *Grand Hotel* dates back to 1864 and is situated on the seafront overlooking the English Channel, close to the famous Brighton 'Lanes'. The hotel boasts modern and purpose-built conference facilities.

The 1996 conference will, as always, feature talks from leading international anti-virus researchers, and from other people with experience in the area. Papers are now invited from anyone with interest and expertise in any computer virus-related field, on technical matters as well as concerning the effects and control of computer viruses within organisations, the legal problems presented by the spread of viruses, or any other topic of relevance to viruses in today's computing environment.

If you are interested in presenting a talk or chairing a discussion group or workshop, an abstract of 50–100 words should be sent to *Virus Bulletin* by Friday, 26 January 1996.

For more information on any aspect of *VB '96*, or to submit an abstract, please contact the Conference Manager, Petra Duffield at the *Virus Bulletin* offices; Tel +44 1235 555139, Fax +44 1235 531889, email pd@virusbtn.com ∎

IBM PC VIRUSES (UPDATE)

The following is a list of updates and amendments to the *Virus Bulletin Table of Known IBM PC Viruses* as of 21 November 1995. Each entry consists of the virus name, its aliases (if any) and the virus type. This is followed by a short description (if available) and a 24-byte hexadecimal search pattern to detect the presence of the virus with a disk utility or a dedicated scanner which contains a user-updatable pattern library.

Type Codes

C	Infects COM files	**M**	Infects Master Boot Sector (Track 0, Head 0, Sector 1)
D	Infects DOS Boot Sector (logical sector 0 on disk)	**N**	Not memory-resident
E	Infects EXE files	**P**	Companion virus
L	Link virus	**R**	Memory-resident after infection

Ahav

CN: An appending, encrypted, 377-byte direct infector. It infects one file at a time and contains the texts '[AHaV]' and 'Gothmog/DHA'.

```
Ahav                8B96 2902 8DB6 0F01 EB07 AD33 C2AB E2FA C3B9 8B00 8BFE EBF2
```

Barrotes.1463

CER: An appending, 1463-byte new member of the Barrotes family. It contains the encrypted message: 'ViRUS de G.D.R. (c) PutoSoft, NO HAY NADA COMO G.D.R. ++VERDAD?? ;-)'. In the samples investigated, the trigger routine overwrites the MBS, displaying the text quoted and scrolling a whole screen from right to left. It triggers on the 34th (sic!) day of a month and thus, of course, is never run.

```
Barrotes.1463       B8FF EECD 213D EEFF 7503 E9DD 0006 B821 35CD 212E 891C 2E8C
```

Cordobes

ER: An appending, polymorphic, 3334-byte virus containing the text: 'C:\AUTOEXEC.BAT' and '@Echo Virus "EL MOSTRO CORDOBES". @Echo No tema por sus datos. Que pase un buen dia. @Echo. @Pause'. The template below detects it in memory.

```
Cordobes            80FC 4E74 1380 FC4F 7437 3D00 F074 052E FF2E 4C01 B8FF FFCF
```

Coyote

ER: An appending, 1103-byte virus, containing the plain-text strings: 'Ipan in xiktli meztli' and 'Coyote'. The latter, which is placed at the end of files, serves as an infection marker. When active in memory, the virus sets the byte at address 0000:033Ch to 43h.

```
Coyote              A186 002E A35E 000E 1FBA A501 B821 25CD 2126 A15C 002E A360
```

Dagger

CN: An appending, 483-byte direct infector containing the plain-text string: '*.* *.COM' and an encrypted signature: 'DaGGER 2'. The payload, which triggers on Tuesdays, includes corruption of CMOS data.

```
Dagger              B42C CD21 80F9 287F 2933 DBD0 8FDD 0243 83FB 0975 F68D 16DD
```

Dex

CER: A stealth, appending, 1356-byte virus which marks an infected file with the signature 'dex' located at the end of its code. It contains the text: '.COM.EXEv08' and 'PKZIP.EXELHA.EXEARJ.EXE'. When active in memory, the 'Are you there?' call (AH=0DEh, Int 21h) returns value DEADh in the AX register.

```
Dex                 B4DE CD21 3DAD DE75 02EB 588C D848 1E8E D8B8 A700 2906 0200
```

ExeHeader.VVM.204

ER: A 204-byte virus which stores its code in the headers of EXE files. It assumes that the address of the original BIOS Int 13h entry is stored at the location 0000:07B4h, and hooks it by changing that value. The virus contains the text: '(c)VVM*'.

```
ExeHeader.VVM.204   BEB4 07BF E602 A5A5 8C4C FEC7 44FC 7302 2E8E 1E2C 0033 F6AD
```

ExeHeader.VVM.205

ER: A 205-byte virus which stores its code in EXE file headers. It hooks Int 13h by using Int 2Fh, AH=13h. The virus contains the text: '(c)VVM'.

```
ExeHeader.VVM.205   B413 BA75 028B DACD 2F2E 8916 E702 2E8C 1EE9 022E 8E1E 2C00
```

ExeHeader.VVM.205.B

ER: A 205-byte virus which stores its code in EXE headers. It hooks Int 13h by changing a value at address 0000:07B4h. It contains the text: '(c)VVM*', and can be detected by the ExeHeader.VVM.205 template.

ExeHeader.XAM.207

ER: A 207-byte virus, related to the VVM family, which hides its code in EXE file headers. It contains the text 'XAM'.

```
ExeHeader.XAM.207   B813 35CD 21BA 7802 B425 CD21 8BD3 BBF4 028C 4F04 8C4F 088C
```

ExeHeader.222

ER: A 222-byte virus which installs itself into the headers of EXE files. It hooks Int 13h and infects files when written onto a disk. All infected programs become files of COM structure.

```
ExeHeader.222       1EB4 138D 1E1A 018B D3CD 2F1F 891E 5901 8C06 5B01 8D16 DE01
```

ExeHeader.Bug.324

ER: A 324-byte virus which hides its code in EXE file headers. It is memory resident in the Interrupt Vector Table area, and directly infects the program: C:\DOS\SMARTDRV.EXE. It contains the text: 'Header.Bug'.

```
ExeHeader.Bug.324   BE4C 0056 A5A5 5FB8 6202 AB33 C0AB BB03 00BD 0500 B810 4A50
```

ExeHeader.Ming.360

ER: A 360-byte virus which hides its code in EXE file headers. When active in memory, it installs itself in the Interrupt Vector Table. It contains the texts ‘Written by Crazy Lord (Ming)’ and ‘ Made in Hong Kong’.

`ExeHeader.Ming.360  31C0 8ED8 B8F5 0187 064C 0026 A3FD 0158 8706 4E00 26A3 FF01`

Gnida

CER: A primitive, overwriting, 71-byte virus infecting any open files (even non-executables).

`Gnida  891D 8C45 02B8 2125 BA1A 01CD 21BA 4701 CD27 80FC 3D74 05EA`

Grob

CER: An overwriting, 2048-byte virus of EXE file structure containing the plain-text string ‘ANARCHY’ and another, encrypted, string: ‘USSR,1994MZ’,’ANARCHY’.

`Grob  BF89 02B8 8002 8706 8400 AB8C C887 0686 00AB B401 9CFF 5F48`

Hypervisor.3120

CER: An appending, stealth, 3120-byte DOS virus which appears to be targeting *NetWare*. When a user with supervisor rights logs on to a network from an infected workstation, the virus creates a new supervisor account under the name of ‘HYPERVISOR’. It contains the encrypted text: ‘HYPERVISOR’, ‘SECURITY_EQUALS’, ‘GROUPS_I’M_IN’, ‘PASSWORD’, ‘IDENTIFICATION’, ‘LOGIN_CONTROL’, and ‘The Hypervisor’. Minor variants can be detected using the same template.

`Hypervisor.3120  3E8B A69A FEFB 061F 2EFF AE9C FE33 C08E D881 2E13 0404 0058`

Hypervisor.3128

CER: An appending, stealth, 3128-byte virus based on the 3120-byte variant. It contains the encrypted texts: ‘SYS:SYSTEM/SYS:LOGIN/NET$BIND.SYS’, ‘NET$BVAL.SYS’, ‘NET$OBJ.SYS’, ‘NET$PROP.SYS’, ‘NET$VAL.SYS’, ‘GROUPS_I’M_IN’, ‘PASSWORD’, ‘IDENTIFICATION’, ‘The’.

`Hypervisor.3128  3E8B A699 FEFB 061F 2EFF AE9B FE33 C08E D881 2E13 0404 0058`

Immortal.2185

ER: A stealth, appending, 2185-byte virus containing the text: ‘IMMORTAL (c) 1994 by MW’. The virus code includes a payload (screen effects) triggering on 22 December.

`Immortal.2185  B42A CD21 81FA 160C 7403 E990 00BD 5000 BA00 B0B4 0FCD 103C`

Jerusalem.1024.sUMsDOS

ER: An appending, 1024-byte hack of the Jerusalem virus with the plain-text string: ‘sUMsDOS’. When an infected file is run on any Monday the 16th, the virus displays the message: ‘M.I.T. VIRUS BY GUESS WHO???’ and waits for a keystroke.

`Jerusalem.1024.sUMsDOS  03F7 2E8B 8D11 00CD 21BC 0007 8CC8 0510 008E D050 B8C5 0050`

Kirti

CN: A prepending, encrypted, 2000-byte direct infector. On 7 July it displays the text: ‘Happy Birthday Kirti!’ and on 22 August the message: ‘Happy Birthday Y.S.’.

`Kirti  81C7 0001 8B0E 0801 8A04 81FE 4001 7204 2AC4 8804 2805 4647`

NV.732

CR: An appending, 732-byte virus with the plain-text strings: ‘OMSPEC=’ and ‘[New Virus] (1992)’.

`NV.732  893E 8400 8C06 8600 FBE9 9980 FCBB 7505 FBF9 CA02 0080 FC4B`

Onkogen

CER: An appending, 1683-byte virus containing the text: ‘COMSPEC=’, ‘CHKLIST.MS’. All infected files are marked by the signature: ‘* Virtual Onkogen *’ placed at the end of the code.

`Onkogen  B80F F0BB 0000 B900 0055 CD21 5D3D 6666 750F 81FB 6666 7509`

Paradise

CER: A polymorphic, circa 1900-byte virus requiring machines with at least an 80286 processor. It contains the texts ‘PANTERAP’ and ‘PARADISE LOST’. On 17 July, when an infected file is run, the virus may display some of the following messages: ‘ICON’, ‘LOST PARADISE’, ‘SHADES OF GOD’, ‘GOTHIC’.

`Paradise  E800 005B B978 0531 ??0D 9043 EB00 E2F7`

PS-MPC.Minnie

CN: An appending, 271-byte, PS-MPC related virus. It is a fast direct infector containing the text ‘I am a Minnie Virus [kR]*.com’.

`PS-MPC.Minnie  8B86 2E02 2E8B 8E0F 0281 C112 013B C174 282D 0300 2E89 8612`

Scotch

CN: An appending, 2611-byte direct infector attacking first COM programs in a randomly chosen subdirectory. It reinfects infected files. When an infected program is called, the virus displays the text: ‘Donner un nombre entre 0 et <random number> pour executer le program demandé:’ and waits for a number from a keyboard. If the given number does not match that selected by the virus, control passes to the DOS prompt after displaying the text: ‘Desolé, j’avais choisi le nombre <selected number>’. If the given number matches that selected by the virus, another message is shown and another file infected. ‘Bravo, vous êtes la nouvelle victime du SCOTCH virus!’

`Scotch  802D 6447 E2FA 81C2 3101 B409 CD21 B42C CD21 8A44 0FB4 008A`

Tai-Pan.438.C

ER: An appending, 438-byte minor variant created by replacing part of the original code with zeros. It can be detected with the same template which detects previous variants.

`Tai-Pan.438.C  3DCE 7B74 0D3D 004B 7503 E808 002E FF2E AF00 0E07 CF50 5351`

V.548

CR: An appending, 548-byte virus which marks all infected files with the word 55AAh located at offset 4 from the beginning of the code.

`V.548  AA56 B924 02C3 9C2E FF1E 8D00 C33D 004B 740F 3DB1 4B74 05EA`

VOL.713

CR: An appending, 713-byte virus containing the string: ‘VOL3:’.

`VOL.713  80FC FE75 05BB 0002 9DCF 3D00 4B74 03E9 6701 B802 3DCD 2173`

INSIGHT

A Man with a Mission

Iceland is a country most people associate (mistakenly, we are assured) with frozen tundra, with a night lasting six months a year, with isolation and with an odd language so far removed from its Norse and Germanic roots as to seem almost completely unrelated. Fridrik Skulason, in common with pop singer Björk, has done much to put Iceland on the map in the past few years.

Fridrik, co-owner with his wife of *Frisk Software International*, makers of *F-Prot*, is a man of great dedication, but not just where viruses are concerned: this man immerses himself in whatever project is at hand. An authority on Icelandic genealogy, one of his long-term projects (planned to occupy the next twenty to twenty-five years) is to build a database detailing the ancestry of everyone who has ever lived in Iceland for whom records exist.

He is patriotic about all that is Icelandic, including his mother tongue: ‘Our language has changed very little over the past 1000 years,’ he asserted. ‘It’s the others that have changed.’ His knowledge of traditional Icelandic lore is phenomenal, and his descriptions of ancient methods of preserving shark graphic enough to picture rather too clearly!

Such nationalism is not merely a family thing, but a quality possessed, Fridrik says, by all Icelanders: ‘Genealogy is a national hobby. My father, my grandfather, my father’s brother, and my great-great grandfather have all written books on it. Just a few years ago, I discovered a manuscript written on this subject by my great-great-great-grandfather that was gathering dust in the *National Archive*.’

Mechanically Speaking

This is a world away from viruses and computers – how did Fridrik first become involved with the machines? ‘I think it was back in 1980; I was 16 or 17,’ he explained. ‘Prior to that, I wanted to be a chemist. The school I entered that year had one computer; an old *Commodore PET* with, I believe, 8KB of memory. To cut a long story short, I haven’t been more than ten metres away from a keyboard since!’

In 1982 he began to study Computer Science, and had the distinction of being one of two computer science students in a class of about one hundred with his own computer. In 1983, the University computing centre employed him as a part-time lab assistant, leading later to a full-time post there.

He completed his degree in 1983, using for his dissertation a program he had written; an Icelandic spell-checker: ‘It is a difficult language to learn,’ he said. ‘To give you an idea, we have 32 different forms of the word yellow! The program was quite successful. In fact, we are still selling the *Windows* version. The first commercial program I wrote was in 1987: it calculated results from Iceland’s general elections. Then, in 1988 I wrote a genealogy program, combining my two great passions. I don’t have a lot of time for it now, but look forward to doing more over the next few decades.’

At the same time, he was doing courses in psychology (‘I haven’t finished; I have one course left to do’) and working part-time for *IBM Iceland* as a contract programmer.

‘They didn’t have any 8086 assembly language programmers,’ he explained, ‘and we had this *IBM* mainframe terminal emulator which couldn’t handle Icelandic characters. I had to patch the program to make it work with the ten Icelandic characters which are not in the English alphabet.

‘One day while I was working there – it must have been January 1989 – *IBM Iceland* got hit by a virus; a virus we now call Cascade.1704.Format.A. There was some panic because it was formatting disks. Being the only assembly language programmer around, I got a copy of the virus, tore it apart and wrote a detector and disinfector for it. I decided that this was something really interesting.

‘At about the same time, there was an outbreak of Jerusalem. I managed to obtain a copy of that virus, and wrote a disinfector for it. Then I got a copy of Brain from a colleague in Scotland who had been hit, and so on… I built up a sizeable collection of viruses – that’s how it started.’

Products and Partners

Fridrik’s company now concentrates on its anti-virus product, *F-Prot*. In addition to having distributors in 27 countries, he works in partnership with three other companies; *Command Software* in the US, *Data Fellows* in Finland, and *Percom* in Germany. What exactly is the relationship between the four?

‘It’s quite complex,’ said Fridrik. ‘In Iceland we produce the DOS version and the engine, and produce and distribute the shareware version. Our partners get our resources and build on top. For example, *Command* produces an NLM version using a modified version of my engine, while *Data Fellows* produces an OS/2 version, also using a modified engine.’

The company produced some of the world’s first heuristic technology, and has always been at the cutting edge of development. Techniques currently incorporated in *F-Prot*, he is confident, will work until the virus count is 25,000 or more. He does not see himself developing a *NetWare* product: his main interest lies in development of the scanning engine, which is always ongoing. Changes are even now being made to this engine: redesigning it and the integrity checker, etc. Another important change is replacement of the current virus information section of the program with a full-blown Hypertext virus information program; a stand-alone product.

Fridrik Skulason, one of the world's foremost virus researchers, counts gardening and genealogy among his other interests.

'Much of this redesign,' he explained, 'is driven by the fact that we have had a very important addition to the company; Vesselin Bontchev joined us early in September.' One of Bontchev's main tasks will be to control virus classification, an issue to which Fridrik gives top priority, and on which he has been working with *CARO*.

'Originally it was Vesselin, myself, and Alan Solomon,' he said of the virus classification committee. 'Lately, though, Dmitry (Gryaznov) has been taking over Alan's responsibilities in this area. Well, we are both maybe a bit obsessed with getting things classified and put in their proper place!'

About CARO

Fridrik was one of the founding members of *CARO* in 1990; others were Bontchev, Solomon, Klaus Brunnstein, Morton Swimmer, Michael Weiner (who has since left the anti-virus field and *CARO*), and Christoph Fischer.

'As I see it,' said Fridrik, '*CARO* is the only organization in this area that has done something useful. It's a little hard to call *CARO* an organization – anyone who saw our last meeting would have qualms about that! It is not a formal group; we are not an industry association, nor do we pretend to be one. Basically we're a bunch of guys that enjoy drinking beer and sharing viruses and information on viruses.'

CARO's most significant achievements to date, in Fridrik's opinion, are the work it has done on standardising virus names, and the intercompany cooperation it has encouraged.

What's Happening

Very little has been seen from virus writers lately, Fridrik feels, which is novel or noteworthy: in his view, the only really interesting new technique is the Word macro viruses. He himself is interested in viruses only as an abstract concept: 'I have never written a virus. Why should I?' he asked. 'It's a very common misunderstanding that anti-virus people have to write viruses to test either new techniques or new features of the program. Still, there are no interesting new developments. For example, the most interesting new development in polymorphic viruses is that there *is* no new development; we just get more of the same old stuff.'

Fridrik's virus nightmare scenario is for a major software developer to ship millions of copies of a product infected with a virus so complex that it would take weeks to find an antidote. Such a virus, he fears, would probably not trigger until after the software was in distribution worldwide.

'Of course, it *may* already have happened,' he speculated. 'Something might have been put into *Windows 95*… it *may* start activating on January 1st… I don't know. Maybe.'

Producing the Future

The past couple of years have seen many smaller companies being swallowed up by giant international conglomerates. Relatively speaking, *Frisk Software International* is itself one of those smaller companies. Does he feel at risk?

'There are quite a few companies that have tried to buy us,' he admitted. '*Certus* tried a long time ago – that didn't work out. Later on there was another, much bigger, company, that tried to buy us. The price was insulting. Then recently another company tried to acquire us, offering ten million dollars, but we simply are not interested, for various reasons.

'Contrary to what one *might* believe,' he went on, 'there is no correlation between the size of a company and its resources, and the quality of its products. So, some big companies with well-recognised names have in the past had lousy products – it's natural for them to try and acquire better technology by buying some of the smaller developers.

'An interesting development is that it is becoming impossible for a newcomer to enter this field. A new company could come out with a generic product, integrity checker or heuristic scanner, but there is no way a new company could come out with a new virus-specific product. The curve; the number of viruses already existing – it's too much. What we see is some older companies dropping out; either giving up, unable to keep up with the demands, with the number of new viruses, or being swallowed by other companies.'

Outside the Labs

Of course, Fridrik has a wide range other interests, all of which obsess him in one way or another: books, gardening, genealogy… is this how he fills his free time?

'Free time?!' Fridrik merely laughed. 'The concept sounds interesting – I'm not sure I recognize it myself. Seriously, I don't have any free time, but I do create some which I spend either on my books or on genealogy, or if my wife has threatened to leave me, on her!'

VIRUS ANALYSIS 1

Now We Are_Three

Eugene Kaspersky

The analysis of multi-partite viruses is one of the more complex tasks for the anti-virus researcher. The two infection mechanisms (files and sectors) and the hooking of different interrupt vectors, among other things, lengthens the time required for a full analysis. It is common for authors of multi-partite viruses to have considerable system programming experience – writing a multi-partite virus is a difficult task. They are often skilled programmers, and give specialists a hard job.

Unfortunately, the number of multi-partite viruses grows from week to week. One of these new multi-partites is Are_Three, a 2048-byte, COM/EXE/MBR/BOOT infector named after an English translation of part of the German message inside the virus code: 'All good things are three'.

Fortunately, it is as not as complex as either One_Half, or several of the other multi-partite/stealth/polymorphic viruses. Are_Three's reliable code does not contain a complex polymorphic engine, stealth routines, or any other tricks which would make it difficult to detect and disinfect – but its multi-partism has helped it to spread very quickly.

Execution of Infected File

When an infected file is executed, control passes to the virus decryption loops. In the file, the virus code is encrypted three times, and is decrypted by loops which are executed in turn (see Figure 1, below right): the first decrypts the code of the second, the second decrypts the code of the third, and after execution of the third, the virus code is completely decrypted and ready to run.

The virus uses the same encryption function in all loops; a simple XOR: the first loop XORs in chunks one word long, the second uses two-word chunks, and the third takes three-word chunks. The virus author has little mathematical experience – encryption using XORing does not become more complex by being executed two more times: the result is the same when only one loop is used with a rolling key.

When decrypted, the virus passes control to the MBR infection routine. Are_Three reads the MBR of the hard drive and checks for the virus code to prevent multiple infection. Two identifying words, at different offsets in the sector, are checked. These are EB5Eh at the start of the boot sector, and 33FFh at offset 60h within it.

Then the virus saves 20h bytes of the original MBR code, and writes 20h bytes of the virus bootstrap procedure into that location. Next, it writes the modified MBR and four sectors of its code to disk, starting at head 0, cylinder 0, sector 4. This method of infection allows the use of FDISK /MBR to repair the MBR. As the virus does not corrupt the partition table, this will recover the MBR in its original form.

The virus encrypts its code before saving it to disk, so (with the exception of the virus bootstrap code within the MBR) it is encrypted in sectors as well as files – the same algorithm is used for both. After infecting the MBR, the virus returns control to the host program, and does not install itself into system memory when being executed from an infected file. The encryption algorithm is the same in sectors and in files.

Loading from Infected Disk

While booting from an infected disk, the virus bootstrap routine receives control from the MBR of a hard drive or the boot sector of a floppy. This routine reads the main body of the virus into memory, and jumps to it.

After decryption, the virus reserves 5KB of system memory for its code and data buffers by decreasing the size of system memory (stored in the word at address 0000:0413h), and copies itself into this space.

Are_Three then hooks Int 13h and Int 1Ch, restores the infected sector from which it booted to its original form (by copying 20h bytes of the host sector to their original location), calls the hard drive MBR infection routine, and returns control to the host sector.

The hard drive MBR infection routine is the same as that used during the execution of an infected file. Are_Three does not check whether or not its code is already resident, and may infect disks twice while loading from an infected non-system floppy disk.

Int 13h: Disk Infection

The virus uses Int 13h to infect diskettes. Are_Three infects when it sees a call to read the first sector of the disk (Int 13h, AX=0201h, CX=1, DH=0). If a call is made to

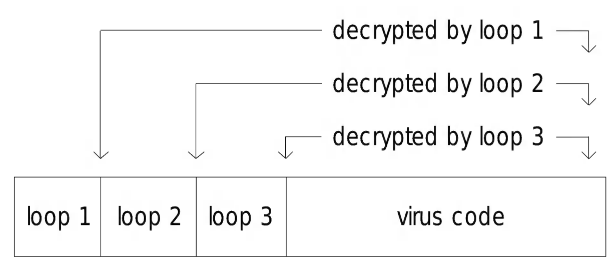


Figure 1: The first loop decrypts the code of the second; the second, that of the third. After the third loop is decrypted, the virus code appears in clear.

Int 13h with any other register values, the virus takes no action other than simply to pass the call on down the interrupt chain.

This virus uses almost the same routine to infect the MBR of the hard drive and boot sectors of floppy disks. The only difference is that the virus checks the disk media byte (at offset 15h within the boot sector), and infects only 5.25-inch 1.2MB and 3.5-inch 1.44MB floppies. Are_Three saves the body of its code in the last four sectors of the floppy. These are liable to be overwritten by data in the course of using the disk, in which case the PC will lock up when it is booted from that disk.

INT 1Ch: Waiting for DOS

The virus uses the technique of hooking Int 1Ch (System Timer Tick) in the same way as many other multi-partite viruses: it 'waits' for DOS to install itself and set its interrupt vectors before hooking Int 21h.

On each call to Int 1Ch, the virus checks the DOS interrupt vectors. If the segment addresses of the Int 20h, the Int 28h, and the Int 2Fh vector handlers are the same, and more than zero but less than 0800h, the virus hooks Int 21h, and stores the Int 2Fh vector address to use later in calls to undocumented DOS functions. During experiments, the virus correctly hooked Int 21h under *DOS v6.x*, but not under *DOS v3.30*.

INT 21h: File Infection

The virus hooks two Int 21h functions: Execute (AX=4B00h) and Open File (AH=3Dh). When either of these is called, the virus calls its infection routine.

First, Are_Three deinstalls the VSAFE/VWATCH anti-virus monitors with a call to Int 16h, AX=FA01h, DX=5945h. It then opens the targeted file and checks its name, infecting only *.CO* and *.EX* files, with the exception of L*.*, -*.*, TB*.*, SC*.*, F-*.*, VI*.* (these are strings which match the names of popular anti-virus utilities).

Next, the virus checks the file date and time stamp to prevent multiple infection – a file is considered infected if the seconds field in the date and time stamp is equal to 26.

The infection procedure continues, branching to handle COM and EXE files differently. The virus reads the file header and checks to see if the first word is the EXE stamp (MZ). Then it increases the file length to align it to paragraph length (10h bytes), encrypts and writes itself at the file end, and overwrites the file header with a JMP instruction in the case of a COM file, or with the new initial CS, IP, SS, or SP register values in an EXE header. When this is complete, the virus exits from the infection routine.

During infection, the virus uses undocumented Int 2Fh calls, and also data from the undocumented System File Table. It also hooks Int 24h to prevent the standard DOS error message while writing to write-protected disks.

Trigger

Are_Three has only one trigger routine: on booting from an infected floppy or hard disk, there is a 50% chance that the virus will display a small red dot, '•', at the top right-hand corner of the screen. Whether this happens depends on the state of the lowest bit of the timer.

The virus also contains the internal text strings:

```
PSYCHo-TECH GMBH 1995
>>> BRAVEd DANGER 4 BRAVE PEOPLe <<<
[[  C·0·D·E·W·A·R  ]]  <31>  Germany  1995
Virtually called to life & survival by MiNDMANiAC!
RGOEPMSQO
==>= AllE GUtEN DiNGE SiND DREi  ==>=
```

Are_Three

Aliases:	Codewar.
Type:	Memory-resident, multi-partite, encrypted, non-polymorphic virus without stealth capabilities.
Infection:	COM/EXE files, MBR of hard drive, boot sector of floppy disk.
Self-recognition in Memory:	None – see analysis.
Self-recognition in Sectors:	Compares first two sector bytes with JMP instruction EBh, 5Eh, then compares two bytes at offset 0060h with 33h, FFh.
Self-recognition in Files:	Seconds field in the file time stamp is set to 26.
Hex Pattern in Files:	FCB9 0104 BD?? ??B8 ???? 813E 2E31 4600 4545 490B C974 03EB
Hex Pattern in Sectors:	33FF BE00 7CFA 8BE6 8ED7 FB8E C7B8 0402 BB00 7EB9 ???? BA??
Hex Pattern in Memory:	3D00 4B74 0A80 FC3D 7405 2EFF 2E59 0660 1E06 9133 C0E8 3302
Intercepts:	Int 21h and Int 13h to infect files/disks, Int 1Ch to hook Int 21h during installation.
Trigger:	Displays red dot at the top right-hand corner of the screen.
Removal:	Under clean system conditions, identify and replace infected files. Use SYS to disinfect floppy disks, and FDISK /MBR to disinfect the MBR of the hard drive.

VIRUS ANALYSIS 2

Satria: It must be Love...

Jakub Kaminski

When one looks at the total number of known viruses, and at the number of those replicating freely in the user community, one finds that while boot sector (including multipartite) infectors account for a small fraction of all viruses, they are responsible for the vast majority of all of the 'in the wild' incidents.

Usually, when a new boot sector infector appears in the real world and not merely in the collections of anti-virus vendors and hobbyists, the odds are pretty good that it will be around for a while.

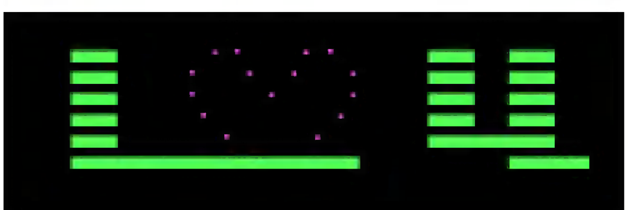
A few months ago, in the middle of May, a new boot sector infector (named after one of its internal strings – Satria) appeared in the wild in Australia. Since then I have received several reports from users who have found disks infected with the Satria virus.

At the end of October, another boot sector virus was discovered at a local university. After a quick look, the Satria we already knew was renamed Satria.A, and a new specimen entered the new family as type .B. The comparison leaves me in no doubt that the two viruses are closely related; however, the differences are quite significant, especially with regard to recovering disks which are infected with Satria.

Welcome, Satria.A

The original Satria (Satria.A) is a one-sector-long virus which infects the Master Boot Sector (MBS) of the hard drive and the DOS Boot Sector (DBS) on floppies. When resident, the virus resides at the top of memory (TOM), and protects itself by reducing the amount of available memory by 1KB – this it does by modifying a value within the BIOS Data Area.

Satria.A contains two characteristic plain-text strings, both of which are visible inside an infected boot sector: 'My Honey B'day' and 'Satria'. Neither message is displayed at any stage by the virus.



Satria.A displays this image when an infected machine is booted on 4 July in any year.

The virus' payload is triggered on 4 July, and consists of a graphic representation of the author's message (a large letter 'I' built from green blocks, a dotted flashing heart, and another letter 'U'). It is for that reason that this virus is sometimes referred to as 'I_love'.

When a PC is booted from an infected floppy, Satria.A lowers the current TOM, copies itself to the reserved area, and hooks Int 13h. Then the virus reads the hidden, original DBS and passes control to it. The new Int 13h service routine intercepts attempts to read the first sector on track 0 of any disk, and infects the media if clean. As a result, Satria.A can infect physical hard disks as well as floppies in both drives.

When the virus infects a hard disk, a copy of the original MBS is written into head 0, cylinder 0, sector 8. As a result of such an infection, IDE drives which contain the extended IDE drivers may become no longer bootable. While infecting a floppy, the virus hides the original DBS in head 1, cylinder 0, sector 3 (in the case of 360KB diskettes) or sector 14 (1.2MB and 1.44MB). This can lead to the loss of a maximum of 16 entries from the root directory. Satria does not infect 720K floppies.

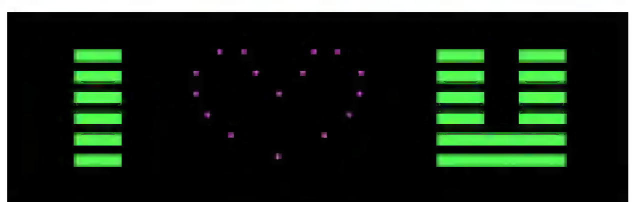
The infection procedure also checks the current system date, and activates the payload on 4 July.

Generally speaking, apart from being a real problem for extended IDE drivers, Satria.A can be seen as quite a harmless, average creature.

Meet Satria.B

Perhaps the author of the original Satria fell so deeply in love that expressing his feelings only once a year simply was not good enough, or maybe someone who was really bored (maybe even the same fellow!) decided to create something new without actually doing much work. The reasons are not very important, but the fact is that we now have to deal with a new virus.

Impatience is the best term to characterise the second variant. When booting from an infected floppy, and after installing itself in memory and activating a new Int 13h procedure, Satria.B does not wait until a hard disk is first



Satria.B's almost identical image is displayed every time the machine is booted from an infected diskette.

accessed, but it checks it straight away and infects if necessary. The original MBS is not stored anywhere on the disk. The partition table is slightly damaged (the last eight bytes of the end of the partition information are lost), and all of the boot procedure is overwritten with the virus code.

Satria.B is also eager to show its message as soon as possible: before passing control to the original DBS, it displays a picture similar to the one in variant .A: this only happens on booting from an infected floppy, so every such boot is almost impossible to miss. Unfortunately, by the time one sees the payload, the hard disk is already infected.

After booting from an infected hard disk, and after installation of the virus code, Satria.B passes control to code loaded from head 1, cylinder 0, sector 1. On most systems, this will work just fine, but systems with a bootable partition starting somewhere else, or disks with additional security systems installed will not start as usual (and will probably crash).

In the case of disks partitioned with standard FDISK and containing fewer than 4 partitions, 'FDISK /MBR' may be used to clean hard disks infected with Satria.B. In other cases, the original MBS has to be restored from rescue/security disks or other clean backup.

Restoring the original DBS of infected floppies is usually impossible. Because of a bug (which is too sophisticated to look like a feature), the virus will reinfect an already infected floppy; thus, a copy of the original DBS will be overwritten forever. Booting from such a floppy will hang the system. As a result, Satria.B infects clean floppy disks (except 720KB), reinfects already infected diskettes, but does not touch floppies originally infected with Satria.A and then with Satria.B. Satria.A leaves its signature (a string 'Satria') at offset 1F7h – Satria.B does not use this area.

Conclusion

It is quite interesting how a type of payload can change the users' perception of a virus. In the reports already received, almost all users were amused and intrigued by the 'nice' screen effect rather than afraid of a new (and potentially dangerous) virus. There is no need to say that most callers saw Satria's payload more than once (some were even trying to trigger it on purpose!).

On the other hand, we all know how easily the word 'Virus' appearing anywhere on a monitor screen, in a text file or CMOS setup can cause panic, sometimes leading even to the formatting of all hard disks. Every anti-virus researcher has his own story about scary messages displayed at boot-up which turned out to be caused by the 'ECHO' command inserted into the file AUTOEXEC.BAT.

Nice-looking things are supposed to be harmless: if something looks and sounds scary, then it must be dangerous. Imagine the impact if Satria, instead of an innocent picture, were to display a threatening message…

Whatever the inspiration was for creating the Satria family, the result, as always, is the same: more viruses in the wild, more for anti-virus products to deal with and (the most important) more potential damage to users' systems.

Satria.A

Aliases: I_love.

Type: MBS/DBS infector. It infects both physical hard disks, but does not infect 720KB floppies.

Self-recognition in MBS/DBS:

The word 00FEh at offset 3Ch.

Hex Pattern in MBS/DBS and in Memory:

```
80FC 0275 1880 FE00 7513 83F9
0175 0E9C 2EFF 1E54 009C E875
```

Intercepts: Int 13h for boot sector infection.

Payload: On 4 July it displays a picture which shows: 'I <flashing heart> U'. Messages inside the virus read: 'Satria' and 'My Honey B'day'.

Removal: MBS – boot clean from DOS floppy. Replace MBS with copy of head 0, cylinder 0, sector 8. Floppy – replace DBS with copy of head 1, cylinder 0, head 1, sector 3 (360KB) or sector 14 (1.2MB or a 1.44MB diskette).

Satria.B

Aliases: I_love.

Type: MBS/DBS infector. It does not infect 720KB floppies.

Self-recognition in MBS/DBS:

The word BFBEh at offset 1Bh.

Hex Pattern in MBS/DBS and in Memory:

```
80FC 0275 1883 F900 7513 83FA
0273 0E9C 2EFF 1E54 009C E8D3
```

Intercepts: Int 13h for boot sector infection.

Payload: Displays picture after booting from infected floppy. Contains no messages.

Removal: MBS – boot clean from DOS floppy. Replace with copy of original sector from backup/rescue disk. If disk has FDISK boot sector, run 'FDISK /MBR'. Floppy – replace DBS with head 1, cylinder 0, sector 3 (360KB) or sector 14 (1.2MB, 1.44MB). If specified sector does not contain a clean boot sector, run 'SYS <drive>' or 'FORMAT <drive>'.

VIRUS ANALYSIS 3

She_Has... Viruses!

Kevin Powis

'She_Has' is more than a virus which is in the wild. It also serves as a warning to remind us all just how easy it is to write a virus with virtually no programming skills.

This virus is a boot sector infector occupying just 361 bytes. When booting from an infected disk (either floppy or fixed), the firmware will load the infected boot sector, just as it would a clean boot sector, into memory at segment 0, offset 7C00h, and pass control to it.

Execution

She_Has commences execution by setting up a pointer to enable it to examine the contents of memory in segment 9C00h. It compares the contents of offset 100h in this segment against the first byte of its own code. If this matches, the virus assumes that it is already resident, so simply loads the original boot sector (as documented below), and passes control to the image of that sector in memory to allow the PC to boot as normal.

If 9C00:100h does not indicate that the virus is already resident, She_Has copies 618 bytes from the start of the virus to its new home in segment 9C00h. Then it reduces a low memory word variable which controls how much conventional memory DOS thinks the PC has. She_Has reduces this word by 16, thereby converting a 640Kilobyte computer to one of 624Kilobytes!

We need to pause here to analyse in detail what has happened so far. First, why does the virus copy 618 bytes when it is only 361 bytes long? There seems to be no logical reason for this. There are, however, clues in the way the virus is structured, which tell us it was written in assembler as a COM file with a fixup of 100h. The snippet of code that decides the number of bytes to be copied looks like this:

```
MOV CX,269h
INC CX
```

Therefore, I think that the author calculated the absolute address of the end of the virus code, which in COM format would have given 269h. Then, as a safeguard which was not actually required, he incremented this to ensure no bytes were left behind.

However, when the virus is translated from COM format into boot sector format, it automatically loses the fixup of 100h bytes, which means 256 bytes more than intended are being copied. If we take 256 from 618 we are left with 362 – the length of the virus body, plus one byte which is the result of the surplus increment instruction above.

With that puzzle behind us, we can ask about another. Why does a 361-byte virus require 16KB of memory? Well, much as I would like to reveal some grand plan, I have to say that this is simply a mistake. The virus author is obviously a novice who does not fully understand memory allocation. By hard-coding in a target segment of 9C00h, he or she has taken the easy option and placed the virus code in the top 16KB of memory before the start of the video RAM.

A possible reason is that the author has been able to calculate manually a fixed target location for the virus, and an associated segment value which will work on any PC of more than 624KB. A more knowledgeable programmer would code a simple algorithm to arrive automatically at the correct segment for stealing 1KB or more from the top of memory.

Leaving our analysis of this programmer behind for a moment, we now continue to follow the virus' execution chain. Once available memory has been reduced, She_Has jumps to the next logical instruction. Instead of staying in the current segment, it passes control to the image of the virus in segment 9C00h.

The next task is to capture the Int 13h vector, which controls all disk access. She_Has constructs a far call using the current interrupt vector contents, which it stores at offset 300 inside its own body. This will give the virus direct access to the ROM BIOS disk handler later. Then it replaces the vector with a pointer to its own interrupt handler, located at offset 136 in the virus body.

With the interrupt handler in place, She_Has reads in the original boot sector, which it has previously saved (see infection) and passes control to it. This allows the PC to boot as normal.

The Interrupt Handler

She_Has then takes control with every disk access, courtesy of its Int 13h handler. On entry to Int 13h, the AH register contains a function value; for example, 02h for read and 03h for write. However, She_Has intercepts only function 00h, which is a disk reset. All other functions are passed on unhindered down the interrupt chain.

When a disk reset is intercepted, She_Has reads the boot sector of the disk concerned, to see whether it is infected. It does this by checking the value at offset 32 in the boot sector: the disk is considered infected if the value is FBh.

Infection

If the boot sector is deemed uninfected, She_Has will first write it unchanged to another sector for later use. When infecting a floppy, the sector used is head 1, cylinder 76, sector 6. On a fixed disk it is head 0, cylinder 0, sector 3.

With the original sector stored away, the boot sector image in memory is patched with a replacement JMP instruction at the start. The virus code is then copied over the top of the clean sector starting at offset 32. This time the virus author remembers to deduct 100h from the number of bytes at first calculated for copying.

Finally, the patched sector is written back to the boot sector of the disk and infection is complete. She_Has allows the interrupted 'disk reset' call to proceed and awaits the next reset function call.

The code for the interrupt handler is structurally the last part of the virus, but there is a visible text message at the end of the virus. It reads 'Virginia / Shirley — She has BREASTS, yes she has !!!'. This message (which is never used) probably says more about the virus author than I ever could.

Conclusion

As I disassembled She_Has, I felt that I was viewing the work of a beginner. A clumsy first attempt. The code has an uncertain feel about it, and there are numerous points which scream out that the author is very inexperienced.

All this aside, whoever he is, he is entitled to have the last laugh because, despite any technical gripes anyone can level at it, the virus works as advertised. It has reached the wild, and infects with ease. It is fortunate that She_Has does not carry any attempt at a trigger or payload.

She_Has

Aliases: Breasts.

Type: Boot sector infector.

Infection: Floppy and hard disks.

Self-recognition:

The offset 32 in any boot sector is equal to FBh.

Hex Pattern: This pattern will locate the virus on hard and floppy disks and in memory.

```
80FC 0074 052E FF2E 2C02 5053
5152 5657 1E06 2E88 162B 02B8
```

Intercepts: Interrupt 13h disk handler.

Trigger: None.

Payload: None.

Removal: The standard method of running FDISK /MBR is sufficient to remove the virus from a hard disk. Removal from floppies can be achieved by salvaging any required files (which will be completely unaffected by the virus) then formatting the floppy.

FEATURE

Worms: An Overview

Glenn Coates, David J. Leigh

In 1975, John Brunner [*who sadly died on 25 August 1995 whilst attending the World Science Fiction Convention in Glasgow. Ed.*] developed the notion of a 'tapeworm' program crawling through a network of computers in his novel *The Shockwave Rider*. This fictional concept bears marked similarities to the worm programs of today. A number of worm programs have already been developed, with the ability to move from machine to machine, utilising system resources to make new copies of themselves on remote machines to which the current host has access.

What is a Worm?

Many users are unfamiliar with the differences between a worm and a virus. A worm is unlike a virus in that it does not need a carrier program in order to replicate. Whereas a virus requires some form of executable code onto which it latches itself, a worm can replicate alone in its entirety.

In simplistic terms, a virus may be thought of as a parasite which must piggyback something in the system which will be run in the normal course of system operations. On the other hand, a worm is an entirely separate 'process' (on multi-tasking machines, each running program is called a process) which uses system resources. In a distributed environment, a worm can exist as a process running on one or more machines which are connected, directly or indirectly, by a network.

A virus can achieve its aims in a single-tasking operating system such as DOS; however, a worm needs the cover of a multi-tasking operating system such as *Windows NT* or *Unix*. Whereas a virus usually requires some form of human interaction to spread from machine to machine (be it transferral by floppy disk or by network) a worm is able to replicate itself without the intervention of a user – it will seek out machines accessible by network and attempt to infect them by introducing and executing copies of itself.

It can be seen, therefore, that a well-designed and well-written worm will often be able to spread much faster within a networked environment than an equally well-designed and well-written virus. It will reach remote machines much faster than the virus, and it will reach machines that the virus never would. Of course, the virus has 'advantages' too – most obviously, it can reach machines with no network path to the source machine, which may ultimately give it a wider distribution.

Systems which permit worms are becoming more popular and are finding an ever-increasing role in the modern corporate world – they have many advantages over the monolithic DOS, and many of these are the same features which render them susceptible to attack by worms. Of itself, this would seem to imply that the prospects for worms are on the increase.

Worm Theory

One of the major papers on the use of worms in a distributed network environment is that by Shock and Hupp [*The 'Worm' Programs – Early Experience with a Distributed Computation. Communications of the ACM 25: 172–180. 1982*].

This paper describes a worm as 'a computation which lives on one or more machines', and goes on to define 'segments' as the programs running on individual computers which make up the worm. In Shock and Hupp's picture, it is not necessary for each segment to consist of the whole worm; in the more general case, a complete copy of the worm can consist of one or more segments.

Should a segment of the worm die, the remaining segments can soon recreate it, effectively 'regrowing' the worm. This regrowth may happen by accident, when another segment happens across the now-uninfected machine and reinfects it; or by design, when another segment detects that one of its fellows has died.

Shock and Hupp created a number of worm programs which they ran on the network at the *Xerox Palo Alto Research Centre*. These tests illustrated that worms were indeed feasible in a beneficial role – some of the uses implemented within *Palo Alto* were an electronic alarm clock (which did not rely on one machine remaining 'up' continuously), and a system to test error rates on a local area network.

It is a sombre footnote to Shock and Hupp's story that one of the first worms they wrote and released got out of control (presumably due to a programming error or program corruption). Fortunately, the authors had demonstrated the type of foresight not later shown by Robert T. Morris (see below) by building into the worm the ability to kill itself in response to a 'kiss-of-death packet'.

Case Study: The Internet Worm

Warnings that it would be possible to create a worm which would take advantage of the Internet were given as early as 1986. Unfortunately, the computing industry took the warnings lightly, and suffered both the direct consequences of the worm and the associated embarrassment.

The Internet Worm was written by Robert T. Morris (better known by his login, RTM), the son of a well known computer security expert, and released on 2 November 1988. Although it had no intended payload, the Worm managed to cripple several sections of the Internet.

The Internet Worm was in some ways a simpler creation that those envisaged by Shock and Hupp (and indeed Brunner). A fully-functioning copy of the Internet Worm always only occupied one segment – that is, each machine carried a complete working copy. It took advantage of various flaws in versions of the *Unix* operating system in use at that time. Once a new segment of the Worm was created, it immediately built a list of remote machines to which it would later attempt to copy itself. This list was created using data from a number of sources on the host machine.

For each host in this list, the Worm attempted to log on in a number of different ways. It was able to gain access at a user level by 'cracking' the password on an account and logging in. It also exploited a bug in the finger daemon (a process which answers requests from remote computers about users of the system), and another flaw in the debug option of the sendmail daemon (a process which handles the despatch and receipt of mail to and from remote computers).

When one of these techniques gave the Worm access to a new machine, it compiled and ran a bootstrap program on that machine by passing commands to the shell. This program then called back to the machine, from which the worm was jumping, to the new host with a request for the rest of its body. The bootstrap program received an encrypted copy of the complete Worm, which was promptly installed. The new segment was then fully-functional and able to proceed to infect machines of its own accord.

In addition to population and growth control, the Worm also utilised various methods of camouflage. It left no traces of itself in the file system, it disabled utilities which enabled administrators to produce memory dumps, and it made extensive use of encryption. It even changed its PID (Process Identifier), so that an administrator would simply think that it was another routine system daemon.

Within hours of its release, the Worm program had spread to thousands of machines attached to the Internet, causing much anxiety amongst system administrators. The day after the Worm's release, fixes were distributed as a counter-attack.

Defences

Dowling showed [*Computer Viruses: Diagonalization and Fixed Points. Notices of the American Mathematical Society*, 37 (7), 858-861, 1990] that if an operating system is complex enough to allow users to copy programs, it is possible for those programs to copy themselves, replicate and spread. Computer viruses and worms are 'an inevitable consequence of fundamental properties of any computing domain'. This is a depressing conclusion: it seems to offer no hope for worm- and virus-free programs. However, it does not say that no defences can be erected and maintained against such software.

The twin facts of the design of the *IBM-PC*, and the enormous amount of 100%-compatible third-party hardware, means that a program carefully written for a PC will run on any of the millions of PCs in use today. However, a worm is no threat to the DOS-based PC, due to the single-tasking and single user nature of the operating system. Such a creature would probably be easily visible.

Of course, the potential does exist for worms to spread under operating systems such as *Unix* (now widely used in client-server and distributed systems); however, it is not that simple. One barrier which *Unix* presents to the development of worm programs, as well as to viruses and other such programs, is the diversity of system architecture, and the different versions of the operating system.

Conclusions

The Internet, which is simply a global network of smaller sub-networks, each with their own trusted and vulnerable hosts, is likely to continue to provide vast and diverse channels for worm infections.

Not all worm programs are bad. With the explosion in the use of the Internet, the future brings a vast array of information services to a user's electronic doorstep. A worm could prove very useful for searching such systems. Acting as a 'gofer', a worm could be used to seek out information in a distributed database.

This technique could also be used in order to process data requests in a hypertext or hypermedia system. Such a worm could 'crawl' through a world-wide network within seconds, do its work, and promptly destroy itself, greatly speeding the work of searching for data [*such systems are already under development. Ed.*].

While the use of DOS has encouraged the spread of viruses, the present growth in the use of multi-tasking OSs such as *Unix*, *OS/2* and *Windows NT* and *95* creates an interesting situation. Such systems are likely to allow worms to become more prevalent. Further, with the increase in high connectivity and distributed systems, an efficient path of communication for worms is created. It is possible that in the future we will be concerned less with polymorphism and traditional stealth mechanisms than with OS flaws and backdoors.

Glenn Coates recently completed a BSc (Hons) degree in Computing Science at *Staffordshire University* (UK). His final year project consisted of developing a prototype Virus Description Language, which he presented in a talk at *VB 94*, and which now has generic decryption capabilities. His latest project is on Neural Networks and Heuristic Scanners. He is currently working as an independent Software Engineer in the north of England.

David J. Leigh, MA Cantab, MSc Nottingham, CEng, FBCS, FIEE, is Director of Studies at *Staffordshire University's School of Computing*. His main specialist area until recently has been compiler writing, and he has also written a book entitled *Software: Its Design, Implementation and Support* (*Paradigm* 1987). He has of late become involved in the fields of computer security and anti-virus research. Leigh can be contacted via email at: D.J.Leigh@soc.staffs.ac.uk.

PRODUCT REVIEW 1

Dr Solomon's Anti-Virus Toolkit for NetWare

Martyn Perry

Dr Solomon's Anti-Virus Toolkit for NetWare v7.5 (*AVTKN*) is the latest *NetWare* product from *S&S*. An earlier version was reviewed by *VB* in May 1994: since then, many changes have been made. This latest version is based on the scanning engine used in the DOS product reviewed in May 1995. This month's review limits itself to evaluating the network features and virus detection.

Presentation and Installation

The *AVTKN* comes with installation disks and documentation for DOS/*Windows* as well as *NetWare*. The fourth edition of the Virus Encyclopædia is also included. The product supports both *NetWare* 3.x and 4.x: different versions have separate installation options. For version 3.x, CLIB 3.12g or later is required: the documentation has information describing how to obtain this from *CompuServe* or the World Wide Web. Without CLIB, the File Access Monitor will not load. Older versions of *NetWare 3.x* may require NWSNUT.NLM.

The installation notes recommend using ATTACH to connect to the target server when performing the initial installation: this will avoid risk of infection from executable files run from the login script. However, it is important to ensure that MAP.EXE is present in the LOGIN directory.

In addition, protection for the server during the boot sequence is included, by making use of the checksum utility (ViVerify) included with the DOS product, and by adding REMOVE DOS to the server's AUTOEXEC.NCF file.

Installation on the server can be performed using the INSTALL command and selecting the version of *NetWare* required – one wonders why the install process cannot detect which version of *NetWare* is required. The 'What's new' manual contains a list of the installed files to allow for manual update or custom batch file installation.

The workstation installation involves copying the VirusGuard files to an appropriate directory and amending its AUTOEXEC.BAT. There are comprehensive installation notes covering installation on diskless workstations. The final requirement for installation is to copy the Configuration Editor files to a suitable *Windows* directory.

Product Features

The product consists of five main elements: a *NetWare* version of FindVirus, a scheduler for timed scans of the server, a File Access Monitor which intercepts file access requests and passes them to the scanner if the file is to be scanned, the VirusGuard TSR for workstation protection, and a *Windows* program to edit the configuration files.

FindVirus can be controlled from the command line by using LOAD NFINDVIR to give on-demand scanning. Alternatively, it can be launched and controlled from another NLM – other NLMs in the system also work like this.

The scheduler provides timed scanning of the server, allowing the server and the workstations to be scanned for viruses at specified times or intervals. Individual scheduled jobs specify a time, what to scan, and a list of whom to notify in the event of trouble. The scheduler can also respond to events on both servers and workstations: this includes new workstations logging on to the server, new volumes being mounted by the server, or virus alerts sent from workstations. This component has four monitors for tracking various activities:

- the Connection Monitor, which watches workstations as they log in to the server, to ensure that they are running VirusGuard
- the Volume Mount monitor, which tracks volumes as they are mounted on the server, and runs *NetWare* FindVirus on any new volume
- the File Access Monitor, which checks if a file should be scanned, and, when the scan has been performed, passes results to the Response Tables (these are lists of actions to be taken in any given circumstance). This list is compared with the result of an event to determine what action needs to be taken. The actions can include: write to log, write to console, broadcast to a notification list, hold an event, clear the current connection, and log out from all servers.
- the Workstations Alert Monitor, which displays alerts from workstations when VirusGuard detects a virus

The Scheduler Main Display shows work pending and currently in progress.

When the scheduler starts, it reads the configuration information into memory. This schedule is repeatedly polled for current events. Each is handled separately from start to finish – results logged, warnings broadcast, and errors cleaned up. This allows several events to run concurrently.

The scheduler is controlled by an ASCII text configuration file residing in the same directory as the scheduler. Multiple versions of this file can exist, and the file required can be specified when the scheduler is started. The configuration file contains a list of items for the scheduler to perform. A default, NTOOLKIT.CFG, is supplied on the install diskettes.

The CFG file must be closed *and* the scheduler unloaded and reloaded in order to read the updated CFG file. It would have been useful if the NLM had been able to detect modifications to the CFG file and update itself accordingly.

The File Access Monitor is an on-access scanner with two components; monitor and scanner. The monitor, FAM.NLM, intercepts server-based file access requests and, if the file is to be scanned, passes it to the scanner. If the file has a virus, FAM.NLM passes the result to the Scheduler to be handled according to the options set in NTOOLKIT.CFG. The scanner, SCANNER.NLM, provides the same facility as the on-demand scanner controlled by the scheduler: it informs FAM.NLM whether or not a file is infected.

"one weakness is that there are no facilities for cross-server administration"

The workstation TSR, VirusGuard, can detect both file and boot sector viruses. Files are scanned when executed or copied by the workstation, both on server and on workstation volumes. VirusGuard intercepts access requests to files on the server and passes their names to the scanner if the file is to be scanned. The network-aware version of VirusGuard not only reports the detection of viruses, but can also act on instructions from the scheduler.

The Configuration Editor, NTKEDIT.EXE, allows a scheduled event to be created or amended and the result stored in the configuration file (which is usually NTOOLKIT.CFG). Although this is a standard ASCII file, using the editor makes the generation of the configuration files much easier, since the user is presented with the majority of the available options using a *Windows* dialog to produce the necessary NFINDVIR command options. The alternative is to use a normal text editor, and create the configuration file manually.

Configuration and Administration

The scheduler NLM is loaded by typing LOAD NTOOLKIT at the console. A command-line option, /C=filename, allows a configuration file other than the default NTOOLKIT.CFG to be specified.

The scheduler displays the next five events due to run, each of which comprises two lines. The first consists of schedule entry number (as generated by the scheduler), name of the event (as specified in the configuration file), the target (server or workstation), and the day and time the event will be run. The second displays the command to be executed.

In addition, a separate window shows the details of any events currently running. This comprises schedule entry number, program or module name, and status code.

Other options include viewing all items in a schedule, and monitoring connections, volume mounts and workstation alerts. It is also possible to view an event log. To prevent unauthorised unloading of the current scheduler session, a password can be set which must be given to unload the NLM.

NetWare FindVirus allows command-line options to be given in a parameter file (default NFINDVIR.INI) to control what is to be scanned and actions to take in case of alert.

Virus Detection

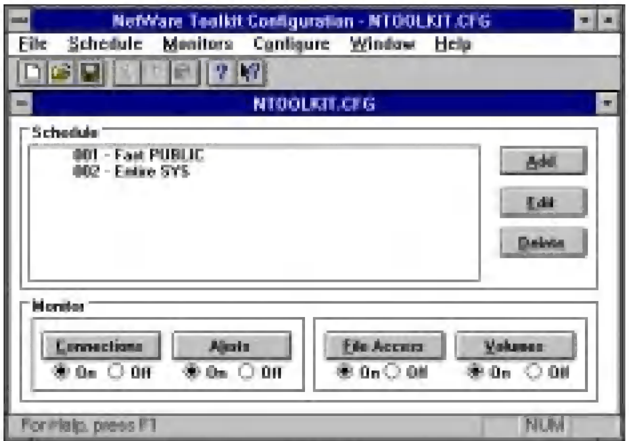
When a virus is detected, several actions can be taken:

- send a message to the application log – this is the Scheduler's log file
- send a message to the console
- broadcast a message to the notification list
- set the hold flag for this event – this event will not run again until cleared from the server console
- disconnect the workstation from the current server. This occurs irrespective of activity on the workstation and could lead to unpredictable results.
- disconnect workstation from all servers to which it had been connected

In addition, the scanner can be configured to perform one of the following actions: attempt to repair an infected file, copy an infected file to a backup directory and attempt a repair on the original, move the infected file to a backup directory, or report only the infected file.

When run against the *Virus Bulletin* test-set, the scanner detection rate was 100% on both the In the Wild and the Polymorphic test-sets. However, the blot on the copy-book was Cruncher in the Standard test-set, one of two samples not detected using the default options. When Pack and Unzip options were used, the detection rate was 100%. *S&S International* states that later versions of the product will scan inside Diet-compressed files by default.

Unfortunately, documentation for the network version lists the switch options available for the *NetWare* version, and makes no reference to Pack and Unzip. It also states that 'all other switches have been removed from the *NetWare* version'. This may leave someone who is unfamiliar with the DOS switches available to deal with viruses in compressed files unnecessarily exposed. The scanner can cope: a simple addendum to the manual would resolve the problem.



Screen shot of the Schedule Event screen from the Configuration Editor.

Real-time Scanning Performance Overhead

The overhead of the real-time scanning functions of the File Access Monitor NLM were tested by copying 136 uninfected COM and EXE files totalling 9,870,393 bytes from the workstation to the server, and comparing the time taken with and without real-time scanning active.

When the scanner was not active, copying took 5 minutes and 6 seconds. When real-time scanning was enabled, the time rose by 1 minute and 20 seconds, to 6 minutes 26 seconds, an increase of 26%. However, it should be noted that the server used is extremely underpowered: in most real-world scenarios, the overhead will probably be less.

Conclusion

The documentation is generally clear and informative, apart from the earlier comment on scanner options. The software is highly configurable, to meet different situations. A licence per server is needed irrespective of the number of users.

One weakness, for a server-based product, is that there are no facilities for cross-server administration. Although existing *NetWare* utilities can be used, it would be helpful to have the facility to monitor other servers and control scanner updates. The updating of scanners seems very much a ‘Cinderella’ activity in many organisations, due to the work involved. Any facility which eases this workload would be highly beneficial and would leave larger organisations less open to errors of omission in the scanner update procedure.

The issue of scanner overhead should not detract from the product’s overall performance, since the scheduler and the options can be configured to meet the requirements of the network. This needs to be considered alongside the excellent detection rate expected and obtained from the *Toolkit*.

Is it better to have a scanner which is fast but has poor detection, or a scanner which works hard to provide the level of protection needed by today’s networks?

I know what my choice would be.

Anti-Virus Toolkit for NetWare

Detection Results

Main scanner:

Standard Test-Set[1]	232/233	99.6%
In the Wild Test-Set[2]	160/160	100%
Polymorphic Test-Set[3]	2500/2500	100%

Overhead of Scheduled Scanning

Time to copy SYS:PUBLIC

Without scanner:	162 seconds
With scanner:	227 seconds (an increase of 65 seconds, or 40%)
Scanner & /slow=10000	175 seconds (an increase of 13 seconds, or 8%)

Technical Details

Product: *Dr Solomon’s Anti-Virus Toolkit for NetWare v7.52.*

Developer/Vendor (UK): *S&S International plc*, Alton House, Gatehouse Way, Aylesbury, Buckinghamshire, HP19 3XU, UK. Tel +44 1296 318700, fax +44 1296 318777. Email:sales@sands.co.uk

Price: One server – £599 (monthly updates), £399 (quarterly); 2–5 servers – £499 (monthly), £299 (quarterly).

Hardware Used: Server – *Compaq 386/20*, 6MBytes Ram, 54MBytes Disk, NetWare 3.11. Workstation – *IBM PS/2 55/SX*, 4MBytes Ram, 33MBytes Disk, *MS-DOS 6.20*, *Windows 3.1*.

[1] Standard Test-set: 1049, 1260, 12_Tricks, 1600, 2100 (2), 2144 (2), 405, 417, 492, 5120, 516, 600, 696, 707, 777, 800, 8888, 8_Tunes, 905, 948, AIDS, AIDS-II, Alabama, Ambulance, Amoeba (2), Amstrad (2), Anthrax, Anti-Pascal (5), Argyle, Athens (2), Armagedon, Attention, Bebe, Big_Bang, Black_Monday (2), Blood, Burger (3), Cascade (2), Casper, Crazy_Lord (2), Cruncher (2), Dark_Avenger.Father (2), Darth_Vader (3), Datacrime (2), Datacrime_II (2), December_24th, Destructor, Dir, DiskJeb, DotKiller, Durban, Eddie, Eddie-2.A (3), Fax_Free.Topo, Fellowship, Fish_1100, Fish_6 (2), Flash, Fu_Manchu (2), Genesis.226, Halley (1), Hallöchen.A (3), Hymn (2), Icelandic (3), Internal, Invisible_Man (2), Itavir, Jerusalem.PcVrs.Ds (4), Jocker, Jo-Jo, July_13th, Kamikaze, Kemerovo, Kennedy, Lehigh, Liberty (5), Loren (2), LoveChild, Lozinsky, Macho (2), MIX1 (2), MLTI, Monxla, Murphy (2), Nina, NukeHard, Old_Yankee (2), Oropax, Parity, Perfume, Phantom1 (2), Pitch, Piter (2), Poison, Polish-217, Power_Pump.1, Pretoria, Prudents, Rat, SBC, Semtex.1000, Shake, Sibel_Sheep (2), Spanz (2), Starship (2), Subliminal, Sunday (2), Suomi, Suriv_1.01, Suriv_2.01, SVC.1689.A (2), Sverdlov (2), Svir, Sylvia, Syslock, Syslock.Macho (2), Syslock.Syslock.A, Taiwan (2), Terror, Tiny (12), Todor (2), Traceback (2), TUQ, Turbo_488, Typo, V2P6, variants of Vacsina.TP (6), Vacsina.Penza.700 (2), Vacsina.634, Vcomm (2), VFSI, Victor, Vienna variants (11), Virdem, Virdem.1336.English, Virus-101 (2), Virus-90, VP, V-1, Warrier, Warrior, Whale, Willow, WinVir_14, variants of Yankee_Doodle.TP (5), Zero_Bug.

[2] In the Wild Test-set: See *VB*, July 1995 p.20.

[3] Polymorphic Test-set: Girafe (500), MtE (200), One_Half (500), Pathogen (500), RDA (100), Satan_Bug (100), SMEG_03 (500), Uruguay.4 (100).

PRODUCT REVIEW 2

VETting for Viruses

Dr Keith Jackson

VET is a virus scanner which claims to be able to provide 'excellent protection against the many viruses trying to infect your PC'. *VET* can be installed on a network, or it can operate on a stand-alone PC running either *MS-DOS* or *OS/2*. As I do not have a network, and do not run *OS/2*, this review necessarily ignores such features.

Documentation

The *VET* documentation comprises one large A5 ring-bound manual, two A5 booklets, and several pieces of miscellaneous bumph. A small section called 'Reviewers Notes' was also included. The main manual is well-written, readable, technically comprehensive, and reasonably well indexed. Even possible compatibility issues with the memory-resident part of *VET* are discussed, a point most products are happy to gloss over. The Appendix, entitled 'Troubleshooting', covers most common problems, and the error messages produced by *VET* are also documented.

The 75-page, A5 publication entitled *Viruses & Your PC* provides a short description of some of the better known viruses (memorably entitled 'The Viral Zoo'!). It contains explanations of what each virus can do, and instructions on how to eradicate viruses. It is clearly written, comprehensive, and easy to follow: one of the better examples of its genre.

One of the booklets provided with the product is entitled 'VET Windows Interface Manual': at just 14 pages long it is rather a bare-bones explanation of the *Windows* software. However, this does not matter very much as the *Windows* part of *VET* contains two *Windows* help files: these explain the *Windows* components, and viruses in general, in detail.

Installation

VET was provided on two 3.5-inch, low-density (720 Kbyte) diskettes, and two 5.25-inch, high-density (1.2 Mbyte) floppies. At least somebody still caters for those poor souls who still use PCs with 5.25-inch floppy disk drives. For both disk types, one floppy disk contained *VET* itself, and the other contained the *Windows* interface software.

Installation can be 'Standard' (the software takes decisions automatically), 'Custom' (information is solicited from the user), or network-aware. I chose the 'Standard' installation, and everything proceeded very smoothly. All that was required from me was a phrase to be used to identify the PC, and to respond to questions asking whether I wanted to make a reference disk, whether the memory-resident software should be installed, and whether an immediate scan of the hard disk should be carried out when installation was complete. The AUTOEXEC.BAT file was automatically updated, although, when using the 'Custom' installation, this will only happen after user confirmation has been sought and given.

Installation amends the PATH statement of the file AUTOEXEC.BAT, and adds two lines of its own immediately after that statement to install the memory-resident component of *VET*, and to perform a scan each time the PC is rebooted. My multi-path boot system fooled *VET's* installation program (as it does so many other such packages), and only the last occurrence of the PATH statement was correctly amended.

Installing the *Windows* parts of *VET* was on the verge of being trivial. When INSTALL.EXE was executed, it asked for the name of the subdirectory in which its files should be stored, produced a very pretty splash screen, and went off to copy/decompress its files. The splash screen contains useful bar graphs showing how much space is left on the various drives, and how much memory is available.

The DOS version of *VET* requires 250 KB of hard disk space; the *Windows* version, 850 KB. The *Windows* version requires the DOS scanner to be installed, so in reality it requires just over 1 MB of hard disk space. The two *Windows* Help files take up 425 KB, so the *VET* executables themselves are compact by comparison. These figures are not profligate when compared with other anti-virus products.

VET's installed files have their date and time stamp altered to the date and time when installation was carried out; therefore, it is impossible to tell at a glance which version is installed. I cannot criticise this enough. If I re-install a product, I want each installation to be the same as the original. *Cybec* states that, although each installed version of *VET* will carry the date and time of that installation, the date/time of the original distribution floppy is unchanged.

Scanning

The *VET* scanner is activated by pressing buttons either to select setup options, or to initiate a scan of a floppy drive, the hard disk, or either of two user-configured scans. Each scan can be either 'Intelligent' or 'Full'. Archive files (PKLITE, LZEXE) are flagged as such during scanning, but *VET* does not seem to scan inside them.

The user interface to this product is different in style from other scanners; however, it is quite simple to use. It is easy to work out when a scan will be complete, as the product maintains an onscreen display of how much disk space has yet to be scanned: this is a neat touch, which other products would do well to copy. What is usually displayed is how much has already *been* scanned, when how much remains to be scanned is infinitely more useful.

The default installation of the DOS version of *VET* scanned the hard disk of my test computer in 1 minute and 1 second (612 files in total occupying 25 MB, of which 299 files were checked). When the *Windows* version of *VET* was used (which invokes the DOS scanner), a scan of the hard disk of my test PC took 1 minutes 12 seconds: an 18% increase over the corresponding DOS scan time reported above.

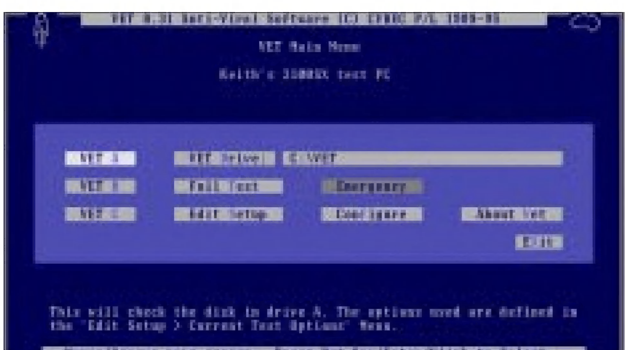
When a 'Full Test' was performed (all parts of each file scanned), scan time rose dramatically, to 5 minutes 2 seconds. It is not possible to scan only executable files when performing a 'Full' scan, as *VET* reports that some options for this type of scan are fixed and 'cannot be edited'. Which files are selected is one such option.

Also, when a 'Full' scan is selected, *VET* does not write the names of all files scanned into the log file, only the names of those files which have been found to be infected by a virus. It is possible, although not immediately obvious how, to configure *VET* to do this. Another annoying habit is that every few lines, seemingly randomly, *VET* inserts a line which just contains three spaces followed by the character ". That's it, no other text is on the line. Most curious.

In comparison, when executing under DOS and performing the same scan, *SWEEP* from *Sophos* took 3 minutes 14 seconds in quick scan mode, and 8 minutes 44 seconds in full mode. *Dr. Solomon's AVTK* took 1 minute 45 seconds to perform the same task in its default operational mode, and 2 minutes 45 seconds when scanning all files. All in all, *VET* is one of the fastest scanners around when used in its default ('Intelligent scan') mode of operation.

VET maintains a thorough log file containing details of what it has found during a scan. Along with other options, the name of this log file can be reset at will. In fact, one of the problems I had whilst using *VET* was that it maintains a different setup for the current scan, a full scan, the scan performed at startup, and alternative scans numbers 1 and 2 (user definable, selectable from the *VET* command line).

The product even maintains a separate set-up for *Windows* execution. This is all highly configurable, but left a mere mortal such as myself somewhat dazed and confused as to what was actually selected at any one particular time.



The main *VET* control screen – unconventional, but eminently usable.

Test-sets

For this review, *VB* provided me with a shiny new test-set. Its content is explained in the Technical Details section. Observant readers will have spotted that I have thus been brought into line with *VB's* normal test-set. I see some logic in being consistent, but it is important to be aware that having just one fixed test-set means products will inevitably be designed which perform brilliantly against this test-set, but perform poorly in the real world. [*It is for this reason that the test-set is revamped at six-month intervals. Ed.*]

VET refused to operate properly with my magneto-optical disk drive. No matter what I tried, it would scan the first ten files, decide that eight of them were infected, and then lock up so thoroughly that a hard reset was required. Other programs do not seem to have these problems when accessing this disk drive. Every time *VET* locked up, it left behind a lost cluster on the hard disk which had to be removed using CHKDSK. Given the content of these files, this seems to be something to do with the *VET* log file.

> *"when tested against the polymorphic samples, VET performed very well indeed"*

All this caused severe problems during testing. The latest *VB* test-set contains a large number of viruses (5000 polymorphic instances) and occupies over 50 MB of hard disk space. I simply do not have room on the hard disk of my test PC to store so many files simultaneously, which is why I usually use the magneto-optical disk for these tests. I had to resort to copying bits of the test-set across to the hard disk to perform scanning tests. This meant it was impossible to test *VET* against all of the more voluminous polymorphic test-sets.

Accuracy

The results obtained when testing *VET* against the Standard and the In the Wild test sets were puzzlingly inconsistent. For instance, when run twice, *VET* often did not give the same result. In fact, there was no need to run *VET* twice; it often did this all by itself!

Several times I started a scan, only for *VET* to stop at a seemingly random point and announce 'Incomplete scan'. This message also appears in the log file. *VET* then restarted the scan, completing it this time. Needless to say, although error messages are documented in the manual, there is no mention of this particular message. Why does this always happen only to me? Funny, that.

I have no idea why all this happens, but clearly it should not. The point at which it occurred seemed random, and the point at which the scan was restarted was also variable, making the measuring of detection accuracy very difficult indeed. Should I simply measure what *VET* did on any one scan, or should a composite result be calculated? The results

quoted below are derived by going through the log files created by *VET*, and finding out if a particular test sample was *ever* detected as virus-infected. This needed lots of detective work, and took ages – the things I do for *VB*!

When run against the test-set described in the Technical Details section, an 'Intelligent' scan detected 106 of the 160 samples in the In the Wild test-set, a rate of just 66%. When a 'Full' scan was invoked, 48 more samples were detected, raising the detection rate to 96%.

This left seven samples of the In the Wild test-set undetected by *VET* – all four of Goldbug, two of Lemming, and the sole Neuroquila sample. The two samples of Anticad and that of HLLC.Even_Beeper contained in the In the Wild test-set were read twice by *VET* and scanned twice. This was another puzzling inconsistency.

Curiously, *VET* performed much better against the Standard test-set than the In the Wild test-set. An 'Intelligent' scan detected all bar seven of the 233 test samples: Aids, Nuke_Hard, and Vacsina.634, and two samples each of Crazy_Lord and Cruncher: a detection rate of 97%.

Although *VET* did not think that the Cruncher test samples were infected, it did state that they were 'Packed by Diet'. Performing a 'Full' scan only detected one more virus (Aids), thus leaving the detection rate basically unchanged.

When tested against the polymorphic samples, *VET* performed very well indeed. It was 100% perfect against 500 MtE samples and 100 Uruguay.4 samples, and missed only one of 256 SMEG samples, one of 256 Pathogen samples, three of 256 One_Half samples, and four of 256 Girafe samples. It also correctly spotted a creditable 98 of 100 Satan_Bug samples. This excellent rate fell to 65% when 100 RDA.Fighter.5871 samples were tested.

My trusty calculator totals this up as detecting 1778 out of 1824 polymorphic virus samples (97%), an excellent result. The results seemed to be the same for both an 'Intelligent' scan and a 'Full' scan.

VET fared very well when tested against boot sector viruses, successfully detecting all 18 test samples. The manual claims that 'VET can keep out all boot sector viruses', and as far as my testing was concerned, this claim does appear to be true. It is obvious that *VET's* developers are keeping up well with the detection of boot sector viruses. Perhaps they are using a generic method?

Windows Components

VET's Windows program acts as a front-end to the DOS scanner, in addition to providing setup and help facilities. Everything works in the usual *Windows* button-pressing manner. The same scanning and setup options are available as were provided with the DOS scanner, with the exception of a 'Full' scan. Somewhat curiously, this button is not present on the *Windows* user interface. I know not why.

VET performing a scan of the test machine's hard drive.

Two buttons on the *Windows* interface give access to 'Virus Information' and 'Help'. Respectively, these provide information on viruses in general, and help with using *VET*: much of the same ground is also covered in the printed documentation. Both are well written, easy to follow, and contain sensible advice rather than the marketing rubbish pushed out by some anti-virus companies. Only 106 viruses are described in detail, but they are the more common ones.

Memory-resident Program

VET's memory-resident component is called VET_RES. When installed, this occupies only 6.9 KB of memory – definitely not excessive. The impact on low memory can be reduced even more by loading VET_RES into high memory. VET_RES cannot produce a warning message whilst *Windows* is executing; it only produces warning beeps, which is something of a failing.

When active, VET_RES looks for just three things: an infected boot sector when a floppy disk is accessed, a call to the *MS-DOS* program load function, and a warm boot. Thus, virus infections are only detected when files are loaded into memory and executed, not when they are merely copied.

The overhead introduced by VET_RES was measured by timing how long it took to copy 40 files (1.25 MB) from one subdirectory to another. Without VET_RES, the files could be copied in 20.6 seconds, rising to 23.0 seconds with VET_RES present. This 12% overhead is minimal; much less than other similar products, and probably undetectable without a stopwatch.

Virus Removal

Whenever *VET* performs a scan, it acts against any viruses detected, extending even to automatic replacement of infected floppy disk boot sectors. Options are available to repair, rename, or delete infected files, but the option of doing nothing does not seem to be provided. At least, I can't figure out how to do it. Why not? The user should decide whether action needs taking, not the developers of *VET*.

In common with my usual practice, I have not attempted to assess how well *VET* can remove viruses from files, and given that a 'do nothing' option was not available, I resorted to using the delete option.

When detecting viruses, *VET* sometimes declares that a file has a virus. More often, however (roughly 75% of the time), it says that a particular file 'may have' the stated virus. The manual claims that the distinction is whether or not a virus is 'Exotic' (a virus believed unlikely to be found in the wild is termed 'Exotic' by *Cybec*).

A batch file called HUNT.BAT is provided, which executes two 'sacrificial goat' programs designed to become infected by ant active viruses. Following execution of HUNT, virus activity can be confirmed further by a program which recalculates checksums for an entire disk.

Such tools are no doubt very useful if an unknown virus is encountered, and are essential for an anti-virus company to have in its armoury. However, I am not sure if they will be either understood or activated by users.

Conclusions

I have reviewed *VET* previously for *VB* (May 1991). Then, I described *VET* as 'impressively quick': the measurements show that it still scans faster than most competing products. The previous version claimed to detect 37 viruses (numbers have increased somewhat!): the results show that *VET* is good at detecting viruses, as long as it uses 'Full' (slower) detection. Particularly outstanding are the boot sector and polymorphic detection rates, both of which are excellent.

VET is now much improved in virus detection, and still one of the fastest scanners around. Note that I said virus detection was *improved*. It could do with more improvement to permit accurate detection and reliable scanning whilst using its fast scanning method. This would permit it to catch up with what the market leaders are currently offering.

Technical Details

Product: *VET*.

Developer/Vendor: *CYBEC Pty*, Suite 3, 350 Hampton Street, Hampton, Victoria 3188, Australia. Tel +613 9521 0655, BBS +613 9521 6109, fax +613 9521 0727. Email info@cybec.com.au

Version Evaluated: Scanner v8.3, *Windows* components v3.53.

Serial Number: None visible.

Price: Includes tech support and quarterly updates (AUS$125 extra for monthly updates). Discounts are available to educational institutions, non-profit organizations, pensioners, and also for renewals: contact *Cybec* for details. Single user licence AUS$126; 2–4 users AUS$178; 5–9 users AUS$298; 10–14 users AUS$698; 15–19 users AUS$698; 20–24 users AUS$848. Larger licences also available on request.

Hardware used: A *Toshiba* 3100SX; a 16 MHz 386 laptop computer with one 3.5-inch (1.4 Megabyte) floppy disk drive, a 40 Megabyte hard disk and 5 Megabytes of RAM, running under *MS-DOS v5.00* and *Windows v3.1*.

Viruses used for testing purposes:

Where more than one sample of a virus is used, the number of samples is shown in parentheses after the virus name. For a complete explanation of each virus, and the nomenclature used, please refer to the list of PC viruses which are published regularly in *Virus Bulletin*.

The Boot Sector test-set currently contains 18 viruses:

AntiExe, BootExe.451, Brain, Empire.Monkey.A, Empire.Monkey.B, ExeBug.A, Form, Italian, Junkie, LZR, Natas, NoInt, NYB, Parity_Boot.B, Peanut, Quox, Sampo, Stoned.NoInt, Unashamed.

The Polymorphic test-set contains 5000 samples of 8 viruses:

Girafe (1050), Groove and Coffee_Shop (500), One_Half (1050), Pathogen (1050), RDA.Fighter.5871 (100), Sat_Bug.Sat_Bug (100), SMEG_v0.3 (1050), Uruguay.4 (100).

The In the Wild test-set contains 160 samples of 77 viruses:

Anticad.4096 (4), Arianna.3375 (4), Avispa.D (2), Barrotes.1310.A (2), BootEXE.451, Butterfly.Butterfly, Captain_Trips (4), Cascade.1701, Cascade.1704, Chill, Coffeeshop (2), CPW.1527 (4), Dark_Avenger.1800.A (3), Dark_Avenger.2100.DI.A (2), Datalock.920.A (3), Diamond.1024.B, Die_Hard (2), Dir-II.A, DOS_Hunter, Fichv.2.1, Flip (2), Flip.2153 (2), Frodo.Frodo.A (4), Ginger, GoldBug (4), Green_Caterpillar.1575.A (3), Helloween (4), Hidenowt, HLLC.Even_Beeper.A, Jerusalem.1808.Standard (2), Jerusalem.Sunday.A (2), Jerusalem.Zero_Time.Austral.A (3), Junkie, KAOS4 (2), Keypress.1232.A (2), Lamer's_Surprise, Lemming (2), Liberty.2857.D (2), Little_Red (2), Macgyver.2803.B, Maltese_Amoeba (3), Necropolis, Necros (2), Neuroquila, No_Frills.No_Frills.843 (2), No_Frills.Dudley (2), Nomenklatura (4), Nothing, November_17th.855.A (2), Npox.963.A (2), Number_of_the_Beast (5), Peanut, Predator.2448 (2), Quicky, Revenge, Riihi, Sat_Bug.Natas, Sat_Bug.Sat_Bug (2), Sayha (2), Screaming_Fist.927 (4), Screaming_Fist.II.696 (2), Stardot.789.D (2), SVC.3103.A (2), Tai-Pan.666 (2), Telecom (4), Tequila.A, Trojector.1463 (6), Trakia.653, Tremor.A (6), Vacsina.TP-05.A (2), Vacsina.TP-16.A, Vampiro, Vienna.648.Reboot.A, VLamix, Voronezh.1600.A (2), Yankee_Doodle.TP.44.A, Yankee_Doodle.XPEH.4928 (2).

The Standard test-set contains 233 samples of 145 viruses:

1049, 1260, 12_Tricks, 1600, 2100 (2), 2144 (2), 405, 417, 492, 5120, 516, 600, 696, 707, 777, 800, 8888, 8_Tunes, 905, 948, AIDS, AIDS-II, Alabama, Ambulance, Amoeba (2), Amstrad (2), Anthrax, Anti-Pascal (5), Argyle, Athens (2), Armagedon, Attention, Bebe, Big_Bang, Black_Monday (2), Blood, Burger (3), Cascade (2), Casper, Crazy_Lord (2), Cruncher (2), Dark_Avenger.Father (2), Darth_Vader (3), Datacrime (2), Datacrime_II (2), December_24th, Destructor, Dir, DiskJeb, Dot_Killer, Durban, Eddie, Eddie-2.A (3), Fax_Free.Topo, Fellowship, Fish_1100, Fish_6 (2), Flash, Fu_Manchu (2), Genesis.226, Halley, Hallöchen.A (3), Hymn (2), Icelandic (3), Internal, Invisible_Man (2), Itavir, Jerusalem.PcVrsDs (4), Jocker, Jo-Jo, July_13th, Kamikaze, Kemerovo, Kennedy, Lehigh, Liberty (5), Loren (2), LoveChild, Lozinsky, Macho (2), MIX1 (2), MLTI, Monxla, Murphy (2), Nina, NukeHard, Old_Yankee (2), Oropax, Parity, Perfume, Phantom1 (2), Pitch, Piter (2), Poison, Polish-217, Power_Pump.1, Pretoria, Prudents, Rat, SBC, Semtex.1000, Shake, Sibel_Sheep (2), Spanz (2), Starship (2), Subliminal, Sunday (2), Suomi, Suriv_1.01, Suriv_2.01, SVC.1689.A (2), Sverdlov (2), Svir, Sylvia, Syslock, Syslock.Macho (2), Syslock.Syslock.A, Taiwan (2), Terror, Tiny (12), Todor (2), Traceback (2), TUQ, Turbo_488, Typo, V2P6, Vacsina.TP.? (6), Vacsina.Penza.700 (2), Vacsina.634, Vcomm (2), VFSI, Victor, Vienna.? (11), Virdem, Virdem.1336.English, Virus-101 (2), Virus-90, VP, V-1, Warrier, Warrior, Whale, Willow, WinVir_14, Yankee_Doodle.TP.? (5), Zero_Bug.

SUBSCRIPTION RATES

Subscription price for 1 year (12 issues) including first-class/airmail delivery:

UK £195, Europe £225, International £245 (US$395)

Editorial enquiries, subscription enquiries, orders and payments:

Virus Bulletin Ltd, 21 The Quadrant, Abingdon, Oxfordshire, OX14 3YS, England

Tel 01235 555139, International Tel +44 1235 555139
Fax 01235 531889, International Fax +44 1235 531889
Email editorial@virusbtn.com
CompuServe address: 100070,1340

US subscriptions only:

June Jordan, *Virus Bulletin*, 590 Danbury Road, Ridgefield, CT 06877, USA

Tel +1 203 431 8720, fax +1 203 431 8165

END NOTES AND NEWS

'Viruses and malicious code replace unauthorized system access as number one concern', according to the results of ***Datapro's* 1995 international user survey** on information security. Jackie Hyde, *Datapro* Security Analyst, reported that only 60% of organizations surveyed have a security policy; a 13% decrease since the 1994 poll. For details on the survey, contact Peter Pacitti, Marketing Communication Specialist, Tel +1 609 764 0100 (ext 2217), fax +1 609 764 2811.

The *British Standards Institution* (*BSI*) has published ***PD 0007*, a summary of *BS 7799***, *A Code of Practice for Information Security Management*, containing the key text of the original, priced at £9.99. The original publication is still available, at £70.00. To obtain either, contact the *BSI* on Tel +44 181 996 7000, fax +44 181 996 7001.

The next round of **anti-virus workshops presented by *Sophos Plc*** will be held in Abingdon, UK, on 17–18 January 1996. Cost for the two-day seminar is £595 + VAT. One session (day one: Introduction to Computer Viruses; day two: Advanced Computer Viruses) can be attended at a cost of £325 + VAT. Contact Julia Line on Tel +44 1235 544028, fax +44 1235 559935, for details.

Infosec, the UK's first dedicated information security show, will be held at the *London Olympia* (London, UK) from 30 April–2 May 1996. **The programme will include conferences and seminars on topical security issues**, and there will be over 100 exhibitors, covering areas including security, virus protection, and access control. Information on attending or exhibiting is available from *Infosec*, Tel +44 181 910 7821.

Reflex Magnetics has scheduled a **Live Virus Overview**. The two-day workshop will take place on 6–7 December at the company's London premises. Costs are: Day One (Overview), £245 + VAT; Day Two (Advanced), £295 + VAT; or both days at £545 + VAT. Details from Rae Sutton on Tel +44 171 372 6666, fax +44 171 372 2507.

Following last month's item in End Notes and News on *McAfee's* decision to distribute **free copies of its anti-virus software to users wishing to scan before installing *Windows 95***, *Symantec* and *S&S International* have announced similar plans in the UK. To obtain these scanners, contact *S&S* on Tel +44 1296 318700 or *Symantec* on Tel +44 1628 592222.

Thompson Network Software has launched another anti-virus package. ***The Doctor Anti-Virus for Windows 95***, which costs US$59.00 (including 12 monthly updates), provides scanning of disks and programs. The program was written specifically for the 32-bit architecture of *Windows 95*. More details from *Thompson*; Tel +1 404 971 8900, fax +1 404 971 8828.

S&S International has released ***Dr Solomon's Anti-Virus Toolkits* for *Windows 95*, for *Windows NT*, and for *SCO Unix***. The packages, which cost £125 each, include various components long familiar to users. For further information on the products, Tel 0800 136657 in the UK and speak to Penny Brennan, or Tel +44 1296 318700 outside the UK. Email sales@sands.co.uk.

SWEEP for Windows 95, from developers *Sophos Plc*, ships from the beginning of December at £295.00 (including 12 monthly updates). The product runs as a continuous 32-bit application, and detects over 7000 viruses. Information can be obtained from Richard Jacobs on Tel +44 1235 544017, fax +44 1235 559935, email rj@sophos.com.

On 8–9 January, 5–6 February, and 4–5 March 1996, *S&S International* is presenting further **Live Virus Workshops** at the *Hilton National Hotel* in Milton Keynes, UK. The two-day course costs £680 + VAT, and offers the opportunity to gain experience with viruses within a secure environment. Contact Julia Bartle for details: Tel +44 1296 318700, fax +44 1296 318777.